A Distributed Data Acquisition System for Real Time Monitoring of Radwaste Storage

Gianfranco Vecchio*1, Sergio Scirè2, Michele Malgeri3, Paolo Finocchiaro4

1,2,4 INFN Laboratori Nazionali del Sud, via S. Sofia 62, 95125 Catania, Italy

3 Dipartimento di Ingegneria Elettrica, Elettronica e Informatica, Università degli Studi di Catania, viale A. Doria 6, 95125 Catania, Italy

*1vecchio@lns.infn.it; 2scire@lns.infn.it; 3michele.malgeri@dieei.unict.it; 4finocchiaro@lns.infn.it

Abstract

This paper describes design and development of a distributed data acquisition system for the monitoring of a critical environment like a radioactive waste storage with the focuses on its hardware and software subsystems, describing the architectures and the technologies used in the system. The application is based on several subsystems (sensors and electronics) developed at INFN-LNS which provides data to be carried into the storage system. In addition, a web service was utilized to handle the storage logic, and different kinds of clients (desktop, web, mobile) were developed to target the exposed services and to inspect the data.

Keywords

Radiation Monitoring; Radioactive Waste; Data Acquisition; Client-Server Systems

Introduction

This paper presents the software and hardware subsystems of a distributed control system aiming at reducing the nuclear risk inside a radioactive waste storage. It has several critical aspects in planning, deploying, maintenance and finally alarms and emergency management. The short and medium term waste is typically stored inside dedicated storage sites, where the prompt detection of any leak, as well as the continuity of knowledge about the drum contents, is very important both for people and environment safety. This paper describes the system developed in the framework of the Detector Mesh for Nuclear Repositories (DMNR) project at INFN-LNS laboratory.

From a technical standpoint, real-time monitoring of radwaste packages could have been already possible years ago, even though with some limitations. Ambient radiation monitors are typically installed in several locations inside the storage sites. These sensors could be replicated in form of a mesh of detectors (i.e. Geiger counters) to be deployed all over the storage site around the packages and interconnected with an overall transmission and control system. However, such a traditional approach would be too expensive, while only providing a rough overall indication about the drums radiation level. Moreover, localized leaks could go undetected still being extremely dangerous for an operator who moves near the leaking package for routine operations. DMNR is a system for online monitoring of short-medium term radioactive waste, based on robust and low-cost components aiming at a tradeoff among cost, performance and reliability.

Such a tradeoff, along with the radiation hardness need, brought us to develop a solution with sensors made of plastic scintillating fibers placed around each drum, readout by means of pairs of Silicon Photomultipliers (SiPM). The plastic fiber is known to be reasonably radiation hard, and it is proved to be capable of counting gamma radiation that comes from the drums. It behaves like a scintillating Geiger-Muller counter whose photosensors are capable of detecting the tiny light signals of few photons. The sensor system, in addition to the scintillating plastic fibers and the Silicon Photomultipliers elements, consists of a counting system based on an FPGA board that handles the data flow coming from the drums and sends it to the higher level of the application.

In the second section, an overview of the monitoring and control system is given. In the third section, the architecture of the whole system has been described with the focus placed on the sensor network, with its lower and higher level components. While in the fourth section the Software Architecture of the system is shown and the server side and the client side of the system are explained. For the first one the choice of web service has been discussed and how the high availability and data redundancy are ensured. For the presentation layer the client developed to show charts and data to the user have been investigated.

DMNR – Monitoring and Control System

In this paper, we will describe in detail the monitoring and control system of the DMNR project. The control system must record full events history and provide users with several tools to analyze the collected data, on both the short and long term. Moreover, it must provide all the relevant details about the radioactive waste packages in store. At any time, it is possible to determine from the database the type, activity, characteristics and history of each single drum stored in each specified location.

Once the data arrive at the higher level section through a redundant network, a robust and reliable software component sends them to an online console for real-time monitoring and to a database system for historical archive. The stored information includes the daily average activity reported by each sensor, any displacement of elements inside the radwaste storage, operator messages, values of detector thresholds, environment temperature and any warning raised by the system. This information is stored inside a relational database to be used by local or remote operators, control authority, civil defense, fire department, etc., according to a defined set of access privileges. Moreover, by means of an online console the above mentioned parameters can be monitored on a real-time basis. Finally, a web application will also be implemented to access the information stored in the database.

We adopted the same philosophy which inspired the whole DMNR project (simple and low-cost but robust and reliable), adopting free and open-source tools and platforms, choosing those with an active community and up to date documentation:

- **Ubuntu Server Edition**: the most popular GNU/GPL-Linux server distribution.
- **Apache Tomcat**: web server and servlet container, which provides a platform for running web applications developed in Java.
- **Apache HTTP Server**: the most common open-source web server.
- **Oracle MySQL**: Relational Database Management System run as a server.
- **Apache Axis2**: core engine to create, publish and consume Web Services in Java and C.
- **Java Enterprise Edition**: enterprise version of Java language for server programming.
- **JavaServer Faces 2.0**: java-based web application framework, based on Model-View-Controller (MVC) architectural pattern, aimed at simplifying the development of user interfaces.
- **DRBD (Distributed Replicated Block Device)**: distributed storage and replication system for Linux platform.
- **Linux-HA (Heartbeat)**: a project for Linux systems that provides high-availability clustering.
- **Hibernate**: a Java persistence framework project that provides a framework for mapping an object-oriented domain model to a traditional relational database.

Architecture

FIG. 1 schematically shows the architecture of the whole system that consists of the sensor network (Scintillating plastic Fibers, Silicon PhotoMultipliers and FPGA boards), the lower level (data acquisition modules, database and logic application layer) and the higher level (local and remote console and data presentation layer).

In order to develop quickly and economically the high speed electronic logic, a Terasic DE1 board has been employed that hosts an ALTERA Cyclone II FPGA module.

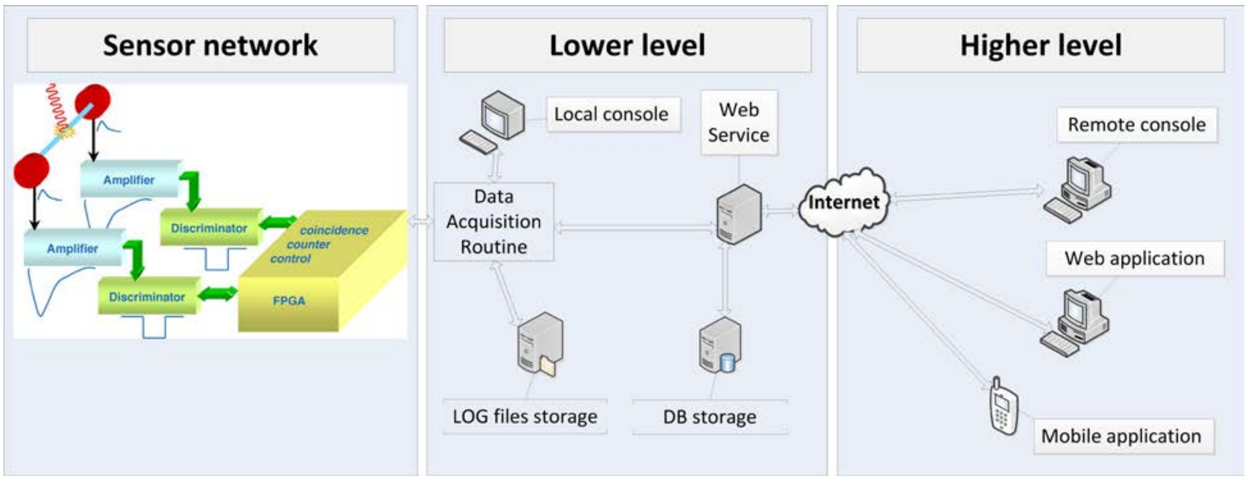


FIG. 1 HARDWARE ARCHITECTURE OF THE WHOLE SYSTEM: SENSOR NETWORKS, LOW AND HIGH LEVEL.

Inside the FPGA, all the needed logic to perform "Data Capture and Dispatch" (DCD) is implemented, along with a UART-compliant transmitter module that serializes the data to be sent. The main communication module, featuring both transmitting and receiving lines, is designed to be easily duplicated into independent communication channels in order to have complementary information pathways and obtain data redundancy at destination. The physical communication medium can be selected between RS-232 and RS-422 standards, the choice depending on foreseen cable length, electrical noise, need of multidrop connection.

All the devices connected inside the lower level make use of the Ethernet cable exploiting a local network (LAN), whereas the data exchanged between the lower and the higher level make use of the internet connection.

Sensor Network

The scintillating fibers employed as radiation sensors in the DMNR system, with polystyrene based core and PMMA cladding, produce a tiny light flash as a result of the interaction of a gamma ray. Each scintillating optical fiber, 1–2 m length and 1 mm diameter, can be arranged around each drum in longitudinal and/or ring geometry. The geometrical efficiency, i.e., the ratio between the fiber solid angle and the 4π total solid angle, is about $d/2\pi r$ where d is the fiber diameter and r is the distance from a pointlike source (when r is much smaller than the fiber length).

The fiber scintillation yield is about 8,000 photons per deposited MeV, but the trapping efficiency only allows the collection at each side of about 4% of the produced photons.

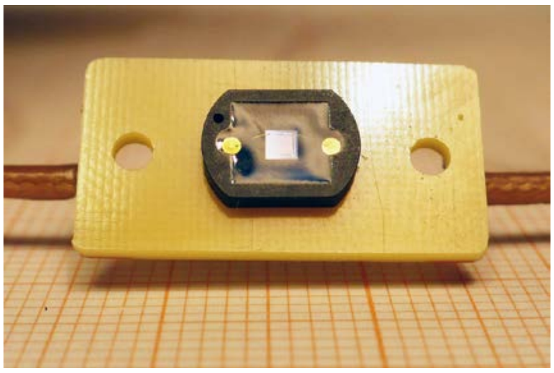


FIG. 2 A 400 MICROCELL SIPM PRODUCED BY SENSL MOUNTED ON PCB. THE REFERENCE GRID PITCH IS 1MM.

In order to detect the few scintillation photons produced by gamma rays, the choice fell on the recently born SiPM (FIG. 2) because of several advantages this technology offers: low cost, low operating voltage, ease of handling, high sensitivity, robustness and reasonable radiation hardness (we estimated 100-1000 years close to a drum with 10-100 mGy/h). This device consists of an array of identical photodiode cells, operated slightly above their breakdown voltage in Geiger avalanche regime, with output terminals connected together.

The sensor assembly is simply made by cutting and polishing fiber ends and then coupled to the photosensor window using optical grease.

The array output signal is the analog sum of those from many independent identical cells, so the SiPM works as a quasi-digital photon counter. Its main problem is the intrinsic dark noise due to cells randomly triggered for thermal reasons. The device is also affected by cross-talk: whenever a cell is triggered, the electrons in the avalanche can themselves generate a few visible photons which have a finite probability to reach and trigger adjacent cells.

In order to make sure that the signal generated by a fiber sensor is actually created by the detection of radiation, it is required that the corresponding light signal overcomes a predefined threshold level and can be detected at both ends of the fiber within a predefined time lapse. Such a time interval can be conveniently tuned according to the maximum time the photons need to run the whole fiber length.

The overall detection efficiency of the fiber sensor, i.e. the fraction of detected gamma rays over the impinging ones, is about 0.001. Such a small efficiency is easily compensated by the possibility of integrating over long time periods, but at the same time it guarantees that the sensor will not be saturated even when exposed to a very high radiation level.

A hardware component (called Time Coincidence Digital Unit TCDU) has been designed in VHDL and deployed into the FPGA, to handle the signals coming from both ends of the fiber and produce the output pulses to be counted, while getting rid of the random dark noise of the SiPM photosensors. TCDU, with two inputs and one output, works as follows: whenever an input pulse is generated by a SiPM, a countdown is started and the circuit waits for the occurrence of another pulse in the other input. If such a second pulse arrives, the output signal is activated, otherwise the circuit is reset and starts waiting again for inputs (FIG. 3).

The duration of the time coincidence window can be tuned quite freely by the FPGA in discrete clock period units down to 10 ns, then fixed at the reasonable value of 50 ns that accounts for any possible difference in the signal pathlength due to the radiation impact position and left/right cable length.

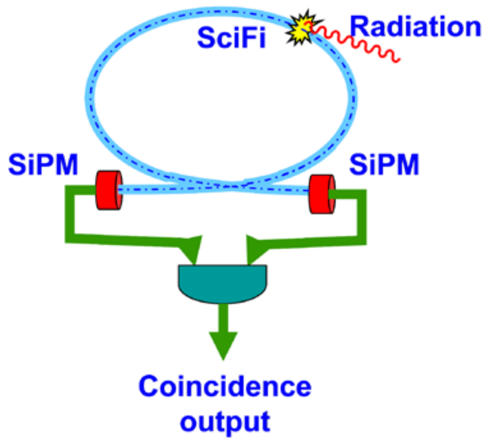


FIG. 3 SCHEMATIC REPRESENTATION OF A SCINTILLATING FIBER-OPTIC SENSOR AND PRODUCTION OF COINCIDENCE ELECTRIC SIGNAL.

Inside DCD, there is a counter module with a dedicated 32 bit buffer for each sensor to be monitored. At each time a sensor gives rise to an output signal, and the corresponding buffer increases its internal count. When a remote operator requires new data, the counter buffer transfers the current count value to a second buffer. The first buffer immediately restarts counting, while the second buffer, after subtracting the previous count value, is transmitted.

The time counter module acts in a similar way but transfers two values: the number of milliseconds elapsed between the current transmission and the previous one (delta time) and the absolute time elapsed from system startup.

In this way it is possible, at destination, to compute the counting rate by simply dividing the delta-counter value by the delta-time one.

A serial frame is generated, for each sensor monitored by the DCD, and each time data need to be sent. The frame is self-consistent, in all the data information it carries for each sensor, allowing at destination detecting if a transmission failure happens (due for example to baud rate misalignment between transmitter and receiver). A start byte allows re-synchronizing the data collection in case of any middle frame byte loss during the transmission. FIG. 4 shows a complete frame that is composed of 14 Bytes. A unique Start Byte is followed by the delta counter value (4 bytes), then the channel number (1 byte), a delta time expressed in milliseconds (2 bytes), the absolute timestamp in milliseconds (4 bytes) and at the end the FPGA identification number (2 bytes).

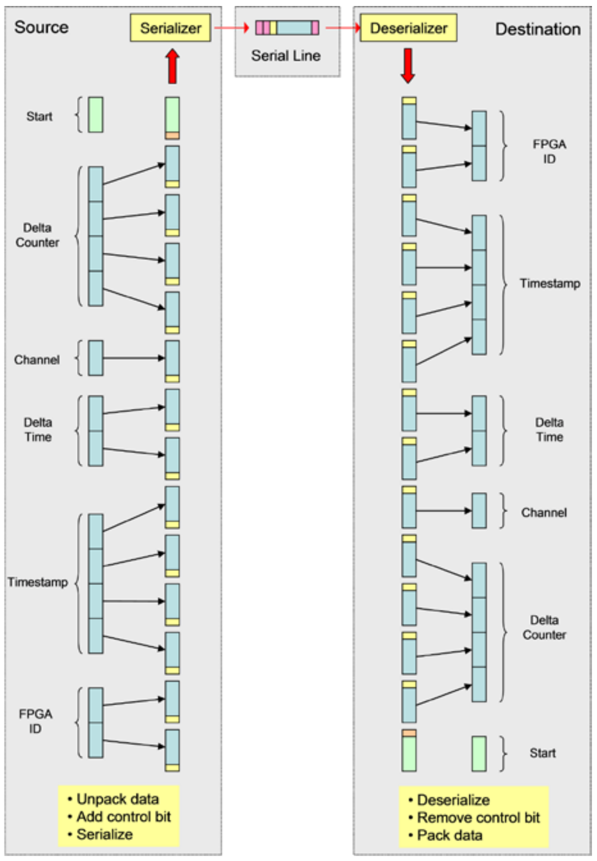


FIG. 4 SERIAL FRAME COMPOSED OF 14 BYTES, COMING FROM EACH MONITORED SENSOR.

The Time Coincidence Digital Unit also implements a UART-compliant transmitter module that serializes the data to be sent. The VHDL main communication module, performing both transmitting and receiving lines, is designed to be easily duplicated into two independent communication channels (for instance one wired and the other wireless), in order to have complementary information pathways and obtain data redundancy at destination.

Lower Level

Data coming from the sensor network are collected by a data acquisition module into a dedicated and robust server machine.

This application is very important for the operation of the whole system because it represents a point of failure of the whole control and monitoring system, therefore we decided to implement Linux-HA and DRBD tools.

However, between the data acquisition routine and the storage system, a web service has been installed that handles both the data entry on the database and the response to client requests, then a web service was selected for its better interoperability among different technologies, code reuse and the capability of data exchange with everyone. The web service includes all the application logic, and exposes all the needed services to insert and retrieve data from the database. Because of the importance of the web service, it is also supported by Heartbeat for high availability and DRBD for data redundancy. The received data packets are sent to the console for online monitoring, while a daily summary is archived into the database, as shown in FIG. 5. A file server stores the log files produced by the data acquisition routine (mirrored with RAID-1 system), while a DB server stores the corresponding retrieval info onto the Oracle MySQL database. All the above described devices are located on a local network and behind a firewall, to prevent any unauthorized access and traffic.

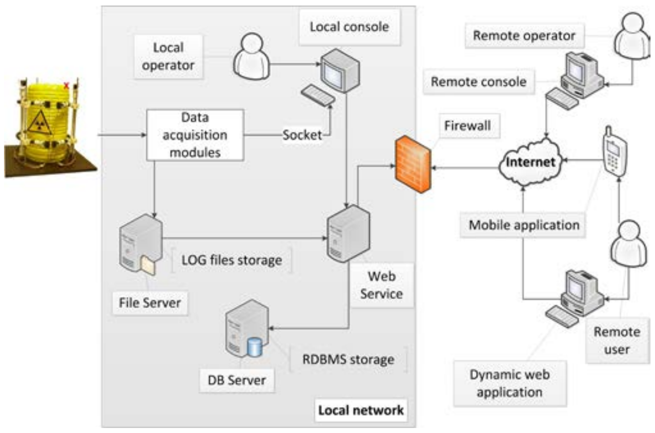


FIG. 5 HARDWARE ARCHITECTURE OF THE LOWER AND HIGHER LEVEL OF THE SYSTEM. IT CONSISTS OF A DATA ACQUISITION ROUTINE, A WEB SERVICE AND A LOCAL CONSOLE FOR THE SERVER SIDE, AND DIFFERENT KINDS OF CLIENTS THAT CONSUME THE WEB SERVICE.

An additional security level was reached through SSL/ TLS security protocol, for confidential communication on the web. The server makes use of a digital certificate to securely identify itself on the web, whereas the clients use an authentication system with credentials.

Higher Level

The higher level provides users with information coming from the sensor system, and handles all controls and warnings. It can produce several categories of warnings/alarms, all of them programmable and configurable.

A local real-time console (LRTC) is developed that shows the status of all the sensors, providing all the needed details about the running monitoring system. LRTC also calculates for each sensor the instant count rate, along with its expected statistical fluctuation; and it also provides several statistical data about the whole system.

Moreover, a web client has been designed and developed to consume the exposed services that has also been optimized for mobile devices.

Software Architecture

The well consolidated three-tier software architecture was adopted to partition the system and allow developers to modify or add a specific layer, rather than to rewrite the entire application. The interaction among all modules is managed by interfaces exploiting the client-server model. The modules are as follows:

1. **presentation layer** dedicated to the user interface,
2. **business logic layer** for the functional logic,
3. **data access layer** for the data storage management.

FIG. 6 shows the detailed software architecture designed and implemented for historical archive of the information.

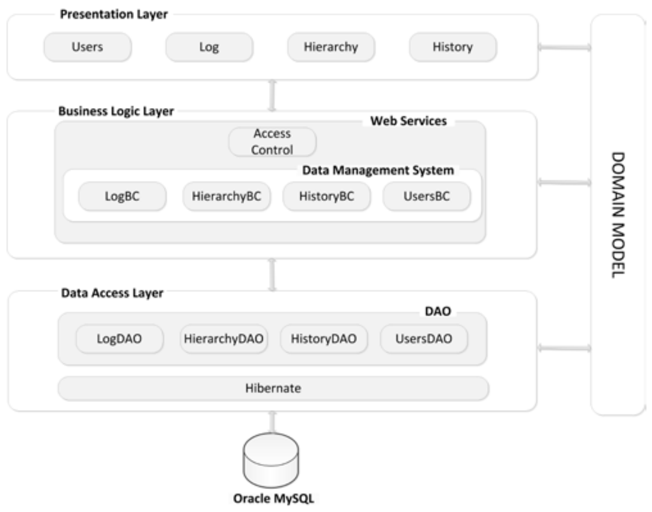


FIG. 6 SOFTWARE ARCHITECTURE OF THE STORAGE SYSTEM. WE ADOPTED A THREE-TIER ARCHITECTURE WITH A DATA ACCESS LAYER, A BUSINESS LOGIC LAYER AND A PRESENTATION LAYER.

The **Data Access Layer** is responsible for providing access to the data stored into the database. This layer uses several Data Access Object (DAO) classes that define functions and procedures to insert and retrieve data to/from the database. In particular, the open source middleware Hibernate has been applied that provides data persistence for Java objects, and generates SQL queries automatically, relieving the programmer from the manual data retrieval and conversion, thus keeping the application portable on all the supported SQL databases.

The **Business Logic Layer** handles the application logic and exchanges information with both Data

Access Layer and Presentation Layer. In our application, it consists of the Web Service and the servlet container Apache Tomcat. They handle users and their roles inside the system, the item hierarchy inside the waste storage, the historical data archive (counting rates and warnings) and the log files produced by the data acquisition system. **Access Control** is an interface between Data Management System and Presentation Layer to handle the user access to the system. **Data Management System** contains the business component classes that provide algorithms to implement the application logic and to call the DAO classes into the Data Access Layer.

Finally, the **Presentation Layer** consists essentially of graphical user interfaces. A local console and a web application have been developed which show forms, charts, tables and all the graphic components to allow the user interaction with the system.

High Availability and Redundancy

To reduce the risk of failure of software or hardware components, a cluster has been built for services supplied by the server and for the data acquisition program. Clustering is a way to ensure high availability by replicating and synchronizing the nodes with each other. Then a solution was designed with two servers (one active, the other passive) that display the same services and are mutually synchronized. To ensure high availability and reliability, Heartbeat, a widely used tool for Linux system, was used; whereas DRBD was selected to replicate data between servers and keep them synchronized.

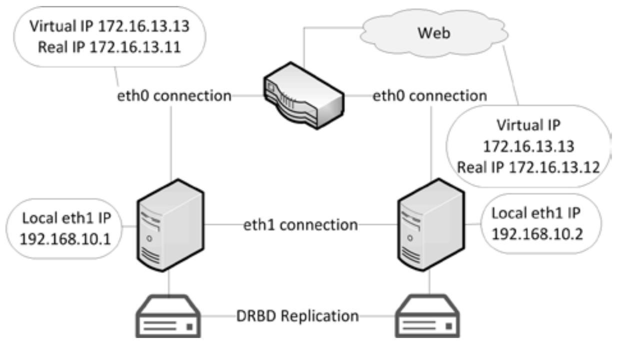


FIG. 7 CLUSTERING CONFIGURATION WITH TWO IDENTICAL SERVERS. WE USED HEARTBEAT AND DRBD FOR HIGH AVAILABILITY AND RELIABILITY.

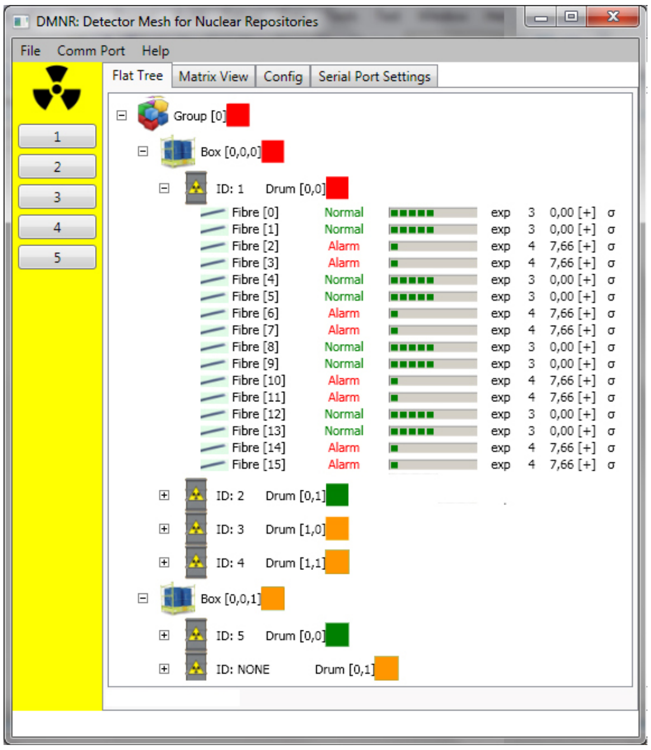
Heartbeat is a system to describe a cluster and then monitor each node to see if it is available. Whenever a node becomes unavailable, Heartbeat can undertake several actions, like moving a floating-IP (the common IP shared between the nodes and used by the active node to respond to an external request), starting or stopping particular services, mounting a particular file system, and fencing the unavailable host (FIG. 7).

A private network is used by the two nodes both to check whether the other node of the cluster is running and to exchange commands and instructions.

The distributed storage system, DRBD, used in conjunction with Heartbeat to replicate and synchronize data between servers is similar to RAID-1, except that it runs over a network. In this manner, more redundancy were added to the main server, like configuration files and database. With DRBD, it is ensured that no transaction will never be lost in case of complete crash of the primary node.

Real-time Console

Using MS .NET 3.0 and C# a console system was designed and implemented reflecting the hierarchical data structure that represents the physical item structure of a radwaste storage site. Whenever a count rate exceeds one of several predefined confidence intervals, the console software raises a corresponding warning.



```
Group [0]
  Box [0,0,0]
    ID: 1     Drum [0,0]
      Fibre [0]    Normal   exp  3  0,00 [+]  σ
      Fibre [1]    Normal   exp  3  0,00 [+]  σ
      Fibre [2]    Alarm    exp  4  7,66 [+]  σ
      Fibre [3]    Alarm    exp  4  7,66 [+]  σ
      Fibre [4]    Normal   exp  3  0,00 [+]  σ
      Fibre [5]    Normal   exp  3  0,00 [+]  σ
      Fibre [6]    Alarm    exp  4  7,66 [+]  σ
      Fibre [7]    Alarm    exp  4  7,66 [+]  σ
      Fibre [8]    Normal   exp  3  0,00 [+]  σ
      Fibre [9]    Normal   exp  3  0,00 [+]  σ
      Fibre [10]   Alarm    exp  4  7,66 [+]  σ
      Fibre [11]   Alarm    exp  4  7,66 [+]  σ
      Fibre [12]   Normal   exp  3  0,00 [+]  σ
      Fibre [13]   Normal   exp  3  0,00 [+]  σ
      Fibre [14]   Alarm    exp  4  7,66 [+]  σ
      Fibre [15]   Alarm    exp  4  7,66 [+]  σ
    ID: 2     Drum [0,1]
    ID: 3     Drum [1,0]
    ID: 4     Drum [1,1]
  Box [0,0,1]
    ID: 5     Drum [0,0]
    ID: NONE        Drum [0,1]
```

FIG. 8 LOCAL CONSOLE GRAPHICAL USER INTERFACE SHOWING THE REAL TIME DATA INFORMATION ABOUT INSTANT RATE FROM EACH SENSOR.

FIG. 8 shows the Graphical User Interface developed in MS Windows Presentation Foundation using DataModel-View-ViewModel (MVVM) pattern which consists of a main panel containing the whole hierarchy of items that show their current state.

Web Application

For the web application, the open source framework *JavaServer Faces* was utilized. JSF simplifies the design and development of a Java web application, without having to mix code and markup. The dynamic web application was created to allow access to the information via web. This application runs on the server under HTTPS protocol, and the client is simply a web browser. Then Java *beans* was employed to manipulate data and call the web services, and Facelets, which replaces JSP, for the creation of views and interfaces.

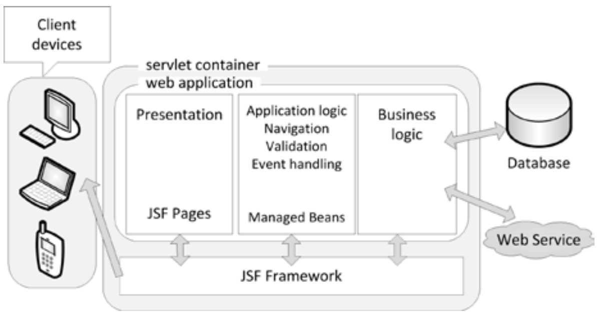


FIG. 9 WEB APPLICATION ARCHITECTURE. IT IS BASED ON THE JSF FRAMEWORK THAT RUNS INSIDE A SERVLET CONTAINER. THE BUSINESS LOGIC OF THIS APPLICATION IS RESPONSIBLE TO CALL THE WEB SERVICE.

JSF, in addition, natively supports event handling and AJAX technology. FIG. 9 shows the architecture of the web application: the presentation layer consists of JSF pages rendered into the client browser, the application logic is made of managed beans, and the business logic calls methods on the web service.

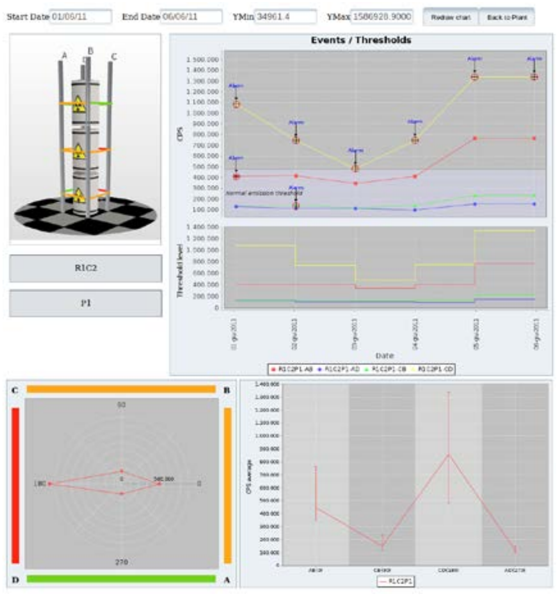


FIG. 10 AVERAGE DAILY COUNTING RATES OF FIBERS, ALONG WITH THE THRESHOLD VALUES OF THE CORRESPONDING SIPM DEVICES. IT IS POSSIBLE TO SHOW DATA FROM A SINGLE FIBER OR ALL THE FIBERS AROUND EACH DRUM.

The Web application can produce a series of data charts to suitably show and navigate through the information stored inside the database. As an example, FIG. 10 shows a chart of the average daily counting rate, along with the threshold values of the corresponding SiPM devices, and the user is free to plot the chart related to a single sensor or to all the sensors surrounding a drum.

Conclusion and Prospects

In this paper, a distributed system has been presented to help operators and control authorities in the real-time online management and monitoring of radioactive waste storage sites. At the moment, the system is finalized, and experimental tests are implemeted in a realistic environment.

Further developments will deal with the study of data correlation between different sensors and drums, in order to build an information-rich 3D-map capable of showing at a glance the overall situation of the whole plant. It is planned to implement an alarm trigger logic exploiting artificial intelligence techniques (neural networks, fuzzy logic, genetic algorithms) in order to establish the more appropriate conditions to raise an alarm.

Moreover, a new design using a redundant Ethernet protocol that in case of fault, removal/insertion of a component will provide deterministic recovery times, is under study.

In the near future, the installation of a pilot system inside a radwaste repository is already scheduled, in order to start collecting experience with a real system in real environment.

ACKNOWLEDGMENT

We are indebted with L. Cosentino, A. Pappalardo, C. Scirè, C. Greco, S. Grillo for their constant support and suggestions. Special thanks to S. Cavallaro, S. Pulvirenti, E. Furia and B. Diana for the fruitful discussions.

REFERENCES

Alur, Deepak et al. “Core J2EE Patterns: Best Practises and Design Strategies (2nd ed.)”, Sun Microsystems Press, May 2003.

Bauer, Christian, and King, Gavin. “Java Persistence with Hibernate”, Manning, November 2006.

Chaurasiya, Vijay. “Linux Highly Available (HA) Fault-

Tolerant Servers", 10th International Conference on Information Technology, December 2007.

Chopra, Vivek et al. "Professional Apache Tomcat 6", Wrox, August 2007.

Crevier, Dan. "DataModel-View-ViewModel pattern series", Microsoft Developer Network Blogs, 2006.

Finocchiaro, Paolo et al. "DMNR: a new concept for real-time online monitoring of short and medium term radioactive waste" in Radioactive Waste: Sources, Types and Management, Nova Science Publishers, in press.

Finocchiaro, Paolo et al. "Silicon photomultipliers for radioactive waste online monitoring" in Nuclear Instruments and Methods in Physics Research A, Elsevier.

Finocchiaro, Paolo et al. "Real-Time online monitoring of Radwaste Storage: A Proof-of-Principle Test Prototype", IEEE Transactions on Nuclear Science, Vol. 59, No. 4, August 2012.

Finocchiaro, Paolo et al. Characterization of a Novel 100-Channel Silicon Photomultiplier – Part I: Noise, IEEE Transactions on Electron Devices, vol.55, (2008) 2757.

Geary, David, and Horstmann, Cay. "Core JavaServer Faces (3rd ed.)", Prentice hall, June 2010.

Kopper, Karl. "Linux Enterprise Cluster: Build a Highly Available Cluster with Commodity Hardware and Free Software", No Starch Press, May 2005.

Monmasson, Eric et al. "FPGAs in industrial control applications," IEEE Trans. Ind. Informat., vol. 7, pp. 224–243, May 2011.

Monmasson, Eric, and Cirstea, Marcian. "FPGA design methodology for industrial control systems—A review", IEEE Trans. Ind. Electron., vol. 54, no. 8, pp. 1824–1842, Aug. 2007.

Perera, Srinath et al. "Axis2, Middleware for Next Generation Web Services", IEEE International Conference on Web Services, December 2006.

Rankin, Kile, and Hill, Benjamin. "The Official Ubuntu Server Book (2nd ed.)", Prentice Hall, August 2010.

Subramanian, Anand, and Melody Ng. "Evaluation of DB2 Universal Database for Linux with DRBD and Heartbeat: A Low-Cost Open Linux High Availability Solution", IBM Toronto Software Lab, October 2005.

Van Vugt, Sander. "Beginning Ubuntu Server Administration: From Novice to Professional", Apress, September 2008.

Vasavi, Bande et al. "Hibernate Technology for an efficient business application extension", Journal of Global Research in Computer Science, Vol. 2, No. 6, June 2011.

Xu JunWu. "Developing CRM System of Web Application Based on JavaServer Faces", Education Technology and Computer Science (ETCS), vol. 1, March 2010.